Diversification improves interpolation

Mark Giesbrecht Daniel S. Roche

April 5, 2011

Abstract

We consider the problem of interpolating an unknown multivariate polynomial with coefficients taken from a finite field or as numerical approximations of complex numbers. Building on the recent work of Garg and Schost, we improve on the best-known algorithm for interpolation over large finite fields by presenting a Las Vegas randomized algorithm that uses fewer black box evaluations. Using related techniques, we also address numerical interpolation of sparse polynomials with complex coefficients, and provide the first provably stable algorithm (in the sense of relative error) for this problem, at the cost of modestly more evaluations. A key new technique is a randomization which makes all coefficients of the unknown polynomial distinguishable, producing what we call a *diverse* polynomial. Another departure from most previous approaches is that our algorithms do not rely on root finding as a subroutine. We show how these improvements affect the practical performance with trial implementations.

1 Introduction

Polynomial interpolation is a long-studied and important problem in computer algebra and symbolic computation. Given a way to evaluate an unknown polynomial at any chosen point, and an upper bound on the degree, the interpolation problem is to determine a representation for the polynomial. In *sparse* interpolation, we are also given an upper bound on the number of nonzero terms in the unknown polynomial, and the output is generally returned in the sparse (also *lacunary* or *supersparse*) representation, wherein only the nonzero terms are explicitly stored.

Applications of sparse interpolation include the manipulation and factorization of multivariate polynomials and system solving (see, e.g., Canny et al. (1989); Kaltofen and Trager (1990); Díaz and Kaltofen (1995, 1998);

Javadi and Monagan (2007, 2009). With the advent of hybrid symbolic-numeric algorithms for (systems of) multivariate polynomials with approximate coefficients, we find applications of approximate sparse interpolation, in particular for solving non-linear systems of equations (see, e.g., Sommese et al. (2001, 2004); Stetter (2004)) and factoring approximate multivariate polynomials (see, e.g., Kaltofen et al. (2008)).

Sparse interpolation is also a non-trivial generalization of the important problem of *polynomial identity testing*: given a black box (especially an algebraic circuit) computing an unknown polynomial, determine whether the polynomial is zero. A relevant result in our setting of sparse polynomials is Bläser et al. (2009); for a more in-depth discussion, we recommend the recent survey by Saxena (2009).

Here we examine the sparse interpolation problem in two settings which have received recent attention: when the coefficients are elements of finite fields (particularly large finite fields, over which we have no choice) and when they are approximations to complex numbers. We give improvements over the state of the art in both of these cases, and demonstrate our new algorithms in practice with a full implementation in C++.

1.1 Problem definition

Let F be a field. A multivariate polynomial $f \in \mathsf{F}[x_1, \ldots, x_n]$ is said to be t-sparse for some $t \in \mathbb{N}$ if f has at most t nonzero terms in the standard power basis; that is, f can be written

$$f = \sum_{i=1}^{t} c_i x_1^{e_{i1}} x_2^{e_{i2}} \cdots x_n^{e_{in}}$$

for coefficients $c_i \in \mathsf{F}$ and exponent tuples $(e_{i1}, \ldots, e_{in}) \in \mathbb{N}^n$ for $1 \leq i \leq t$. If each $e_i < d$, then the size of this representation is $O(t)$ field elements plus $O(tn \log d)$ bits. We seek algorithms which are polynomial-time in the size of this representation.

Let $f \in \mathsf{F}[x_1, \ldots, x_n]$ have degree less than d. A *black box* for f is a function which takes as input a vector $(a_1, \ldots, a_n) \in \mathsf{F}^n$ and produces $f(a_1, \ldots, a_n) \in \mathsf{F}$. The cost of the black box is the number of operations in F required to evaluate it at a given input.

Clausen et al. (1991) showed that, if only evaluations over the ground field F are allowed, then for some instances at least $\Omega(n^{\log t})$ black box probes are required. Hence if we seek polynomial-time algorithms, we must extend the capabilities of the black box. To this end, Díaz and Kaltofen (1998) introduced the idea of an *extended domain black box* which is capable of evaluating $f(b_1, \ldots, b_n) \in \mathsf{E}$ for any $(b_1, \ldots, b_n) \in \mathsf{E}^n$ where E is any extension field of F. That is, we can change every operation in the black box to work over an extension field, usually paying an extra cost per evaluation proportional to the size of the extension.

Motivated by the case of black boxes that are division-free algebraic circuits, we will use the following model which we believe to be fair and cover all previous relevant results. Here and for the remainder, $\mathsf{M}(m)$ is the number of field operations required to multiply two univariate polynomials with degrees less than m, and $O\tilde{~}(m)$ represents any function bounded by $m(\log m)^{O(1)}$. Using Cantor and Kaltofen (1991), $\mathsf{M}(m) \in O(m \log m \operatorname{loglog} m)$, which is $O\tilde{~}(m)$.

Definition 1.1. *Let $f \in \mathsf{F}[x_1, \ldots, x_n]$ and $\ell > 0$. A* remainder black box *for f with size ℓ is a procedure which, given any monic square-free polynomial $g \in \mathsf{F}[y]$ with $\deg g = m$, and any $h_1, \ldots, h_n \in \mathsf{F}[y]$ with each $\deg h_i < m$, produces $f(h_1, \ldots, h_n) \operatorname{rem} g$ using at most $\ell \cdot \mathsf{M}(m)$ operations in F.*

This definition is general enough to cover the algorithms we know of over finite fields, and we submit that the cost model is fair to the standard black box, extended domain black box, and algebraic circuit settings. The model makes sense over complex numbers, as we will see.

1.2 Interpolation over finite fields

We first summarize previously known univariate interpolation algorithms when F is a finite field with q elements and identify our new contributions here. For now, let $f \in \mathbb{F}_q[x]$ have degree less than d and sparsity t. We will assume we have a remainder black box for f with size ℓ. Since field elements can be represented with $O(\log q)$ bits, a polynomial-time algorithm will have cost polynomial in ℓ, t, $\log d$, and $\log q$.

For the dense output representation, one can use the classical method of Newton/Waring/Lagrange to interpolate in $O\tilde{~}(\ell d)$ time (von zur Gathen and Gerhard, 2003, §10.2).

The algorithm of Ben-Or and Tiwari (1988) for sparse polynomial interpolation, and in particular the version developed by Kaltofen and Yagati (1989), can be adapted to arbitrary finite fields. Unfortunately, these algorithms require t discrete logarithm computations in $\mathbb{F}_q^*$, whose cost is small if the field size q is chosen carefully (as in Kaltofen (2010)), but not in general. For arbitrary (and potentially large) q, we can take advantage of the fact that each discrete logarithm that needs to be computed falls in the range

	Probes	Probe degree	Computation cost	Total cost
Dense	d	1	$O\tilde{}(d)$	$O\tilde{}(\ell d)$
Ben-Or & Tiwari	$O(t)$	1	$O(t^2 + t\sqrt{d})$	$O\tilde{}(\ell t + t^2 + t\sqrt{d})$
Garg & Schost	$O\tilde{}(t^2 \log d)$	$O\tilde{}(t^2 \log d)$	$O\tilde{}(t^4 \log^2 d)$	$O\tilde{}(\ell t^4 \log^2 d)$
Randomized G & S	$O\tilde{}(t \log d)$	$O\tilde{}(t^2 \log d)$	$O\tilde{}(t^3 \log^2 d)$	$O\tilde{}(\ell t^3 \log^2 d)$
Ours	$O(\log d)$	$O\tilde{}(t^2 \log d)$	$O\tilde{}(t^2 \log^2 d)$	$O\tilde{}(\ell t^2 \log^2 d)$

Table 1: Sparse univariate interpolation over large finite fields, with black box size ℓ, degree d, and t nonzero terms

$[0, 1, \ldots, d-1]$. The "kangaroo method" of Pollard (1978, 2000) can, with high probability, compute such a discrete log with $O(\sqrt{d})$ field operations. Using this algorithm makes brings the total worst-case cost of Ben-Or and Tiwari's algorithm to $O(t\ell + t^2 + t\sqrt{d})$.

The current study builds most directly on the work of Garg and Schost (2009), who gave the first polynomial-time algorithm for sparse interpolation over an arbitrary finite field. Their algorithm works roughly as follows. For very small primes p, use the black box to compute f modulo $x^p - 1$. A prime p is a "good prime" if and only if all the terms of f are still distinct modulo $x^p - 1$. If we do this for all p in the range of roughly $O(t^2 \log d)$, then there will be sufficient good primes to recover the unique symmetric polynomial over $\mathbb{Z}[y]$ whose roots are the exponents of nonzero terms in f. We then factor this polynomial to find those exponents, and correlate with any good prime image to determine the coefficients. The total cost is $O\tilde{}(\ell t^4 \log^2 d)$ field operations. Using randomization, it is easy to reduce this to $O\tilde{}(\ell t^3 \log^2 d)$.

Observe that the coefficients of the symmetric integer polynomial in Garg & Schost's algorithm are bounded by $O(d^t)$, which is much larger than the $O(d)$ size of the exponents ultimately recovered. Our primary contribution over finite fields of size at least $\Omega(t^2 d)$ is a new algorithm which avoids evaluating the symmetric polynomial and performing root finding over $\mathbb{Z}[y]$. As a result, we reduce the total number of required evaluations and develop a randomized algorithm with cost $O\tilde{}(\ell t^2 \log^2 d)$, which is roughly quadratic in the input and output sizes. Since this can be deterministically verified in the same time, our algorithm (as well as the randomized version of Garg & Schost) is of the Las Vegas type.

The relevant previous results mentioned above are summarized in Table 1, where we assume in all cases that the field size q is "large enough". In the table, the "probe degree" refers to the degree of g in each evaluation of the remainder black box as defined above.

1.3 Multivariate interpolation

Any of the univariate algorithms above can be used to generate a multivariate polynomial interpolation algorithm in at least two different ways. For what follows, write $\rho(d,t)$ for the number of remainder black box evaluations required by some univariate interpolation algorithm, $\Delta(d,t)$ for the degree of the remainder in each evaluation, and $\psi(d,t)$ for the number of other field operations required besides black box calls. Observe that these correspond to the first three columns in Table 1.

The first way to adapt a univariate interpolation algorithm to a multivariate one is Kronecker substitution: given a remainder black box for an unknown $f \in \mathsf{F}[x_1,\ldots,x_n]$, with each partial degree less than d, we can easily construct a remainder black box for the univariate polynomial $\hat{f} = f(x, x^d, x^{d^2}, \ldots, x^{d^{n-1}}) \in \mathsf{F}[x]$, whose terms correspond one-to-one with terms of f. This is the approach taken for instance in Kaltofen (2010, §2) for the interpolation of multivariate polynomials with rational coefficients. The cost is simply the cost of the chosen underlying univariate algorithm, with the degree increased to d^n.

The other method for constructing a multivariate interpolation algorithm is due to Zippel (1990). The technique is inherently probabilistic and works variable-by-variable, at each step solving a number of $\hat{t} \times \hat{t}$ transposed Vandermonde systems, for some $\hat{t} \le t$. Specifically, each system is of the form $Ax = b$, where A is a $\hat{t} \times \hat{t}$ matrix of scalars from the coefficient field $\mathbb{F}_q$. The vector v consists of the output of $\hat{t}$ remainder black box evaluations, and so its elements are in $\mathbb{F}_q[y]$, and the system must be solved modulo some $g \in \mathbb{F}_q[y]$, as specified by the underlying univariate algorithm. Observe however that since A does not contain polynomials, computing $x = A^{-1}b$ requires no modular polynomial arithmetic. In fact, using the same techniques as Kaltofen and Yagati (1989, §5), employing fast dense bivariate polynomial arithmetic, each system can be solved using

$$O\Big(\mathsf{M}\big(t \cdot \Delta(d,t)\big) \cdot \log\big(t \cdot \Delta(d,t)\big)\Big)$$

field operations.

Each transposed Vandermonde system gives the remainder black box evaluation of each of $\hat{t}$ univariate polynomials that we are interpolating in that step. The number of such systems that must be solved is therefore $\rho(d,t)$, as determined by the underlying univariate algorithm. Finally, each

	Kronecker	Zippel
Dense	$O\tilde{}(\ell d^n)$	$O\tilde{}(\ell ntd)$
Ben-Or & Tiwari	$O\tilde{}(\ell t + t^2 + td^{n/2})$	$O\tilde{}(nt^3 + nt^2\sqrt{d} + \ell nt^2)$
Garg & Schost	$O\tilde{}(\ell n^2 t^4 \log^2 d)$	$O\tilde{}(\ell nt^5 \log^2 d)$
Randomized G & S	$O\tilde{}(\ell n^2 t^3 \log^2 d)$	$O\tilde{}(\ell nt^4 \log^2 d)$
Ours	$O\tilde{}(\ell n^2 t^2 \log^2 d)$	$O\tilde{}(\ell nt^3 \log^2 d)$

Table 2: Sparse multivariate interpolation over large finite fields, with black box size ℓ, n variables, degree d, and t nonzero terms

of the $\hat{t}$ univariate interpolations proceeds with the given evaluations. The total cost, over all iterations, is

$$O\tilde{}\big(\ell nt \cdot \Delta(d,t) \cdot \rho(d,t)\big)$$

field operations for the remainder black box evaluations, plus

$$O\tilde{}\big(nt\psi(d,t) + \ell nt \cdot \Delta(d,t)\big)$$

field operations for additional computation. Zippel (1990) used the dense algorithm for univariate interpolation; using Ben-Or and Tiwari's algorithm instead was studied by Kaltofen and Lee (2003).

Table 2 summarizes the cost of the univariate algorithms mentioned above applied to sparse multivariate interpolation over a sufficiently large finite field, using Kronecker's and Zippel's methods.

For completeness, we mention a few more results on closely related problems that do not have a direct bearing on the current study. Grigoriev et al. (1990) give a parallel algorithm with small depth but which is not competitive in our model due to the large number of processors required. A practical parallel version of Ben-Or and Tiwari's algorithm has been developed by Javadi and Monagan (2010). Kaltofen et al. (1990) and Avendaño et al. (2006) present modular algorithms for interpolating polynomials with rational and integer coefficients. However, their methods do not seem to apply to finite fields.

1.4 Approximate Polynomial Interpolation

In Section 4 we consider the case of approximate sparse interpolation. Our goal is to provide both a numerically more robust practical algorithm, but also the first algorithm which is provably numerically stable, with no heuristics or conjectures. We define an "ϵ-approximate black box" as one which evaluates an unknown t-sparse target polynomial $f \in \mathbb{C}[x]$, of degree d, with relative error at most $\epsilon > 0$. Our goal is to build a t-sparse polynomial g such that $\|f - g\|_2 \leq \epsilon \|f\|_2$. A bound on the degree and sparsity of the target polynomial, as well as ϵ, must also be provided. In Section 4 we formally define the above problem, and demonstrate that the problem of sparse interpolation is well-posed. We then adapt our variant of the Garg and Schost (2009) algorithm for the approximate case, prove it is numerically accurate in terms of the relative error of the output, and analyze its cost. We also present a full implementation in Section 5 and validating experiments.

Recently, a number of numerically-focussed sparse interpolation algorithms have been presented. The algorithm of Giesbrecht et al. (2009) is a numerical adaptation of Ben-Or and Tiwari (1988), which samples f at $O(t)$ randomly chosen roots of unity $\omega \in \mathbb{C}$ on the unit circle. In particular, ω is chosen to have (high) order at least the degree, and a randomization scheme is used to avoid clustering of nodes which will cause dramatic ill-conditioning. A relatively weak theoretical bound is proven there on the randomized conditioning scheme, though experimental and heuristic evidence suggests it is much better in practice. Cuyt and Lee (2008) adapt Rutishauser's *qd* algorithm to alleviate the need for bounds on the partial degrees and the sparsity, but still evaluate at high-order roots of unity. Approximate sparse rational function interpolation is considered by Kaltofen and Yang (2007) and Kaltofen et al. (2007), using the Structured Total Least Norm (STLN) method and, in the latter, randomization to improve conditioning. Approximate sparse interpolation is also considered for integer polynomials by Mansour (1995), where a polynomial-time algorithm is presented in quite a different model from ours. In particular the evaluation error is absolute (not relative) and the complexity is sensitive to the bit length of the integer coefficients.

Note that all these works evaluate the polynomial only on the unit circle. This is necessary because we allow and expect f to have very large degree, which would cause a catastrophic loss of precision at data points of non-unit magnitude. Similarly, we assume that the complex argument of evaluation points is exactly specified, which is again necessary because any error in the argument would be exponentially magnified by the degree.

The primary contribution of the work in this paper is to provide an algorithm with both rigorously provable relative error and good practical performance. Our algorithm typically requires $O\tilde{~}(t^2 \log^2 d)$ evaluations at primitive roots of unity of order $O\tilde{~}(t^2 \log d)$ (as opposed to order d in previous approaches). We guarantee that it finds a t-sparse polynomial g such that $\|g - f\|_2 \leq 2\epsilon \|f\|_2$. An experimental demonstration of the numerical robustness is given in Section 5.

2 Sparse interpolation for generic fields

Here and for the remainder, we say a polynomial f is *t-sparse* if it can be written as a sum of at most t nonzero coefficients times a monomial. We assume the unknown polynomial f is always univariate. This is without loss of generality, as we can use the Kronecker substitution as discussed above. The exponential increase in the univariate degree only corresponds to a factor of n increase in $\log \deg f$, and since our algorithms will ultimately have cost polynomial in $\log \deg f$, polynomial time is preserved.

Assume a fixed, unknown, t-sparse univariate polynomial $f \in \mathsf{F}[x]$ with degree at most d. We will use a remainder black box for f to evaluate $f \operatorname{rem} (x^p - 1)$ for small primes p. We say p is a "good prime" if the sparsity of $f \operatorname{rem} (x^p - 1)$ is the same as that of f itself — that is, none of the exponents are equivalent modulo p.

The following lemma shows the size of primes required to randomly choose good primes with high probability.

Lemma 2.1. *Let $f \in \mathsf{F}[x]$ be a t-sparse polynomial with degree d, and let $\lambda = \max\left(21, \left\lceil \frac{5}{3} t(t-1)\ln d \right\rceil\right)$. A prime chosen at random in the range $\lambda, \ldots, 2\lambda$ is a good prime for f with probability at least $1/2$.*

Proof. Write $e_1, \ldots, e_t$ for the exponents of nonzero terms in f. If p is a bad prime, then p divides $(e_j - e_i)$ for some $i < j$. Each $e_j - e_i \leq d$, so there can be at most $\log_\lambda d = \ln d / \ln \lambda$ primes that divide each $e_j - e_i$. There are exactly $\binom{t}{2}$ such pairs of exponents, so the total number of bad primes is at most $(t(t-1)\ln d)/(2 \ln \lambda)$.

From Rosser and Schoenfeld (1962, Corollary 3 to Theorem 2), the total number of primes in the range $\lambda, \ldots, 2\lambda$ is at least $3\lambda/(5 \ln \lambda)$ when $\lambda \geq 21$, which is at least $t(t-1)\ln d / \ln \lambda$, at least twice the number of bad primes. $\square$

Now observe an easy case for the sparse interpolation problem. If a polynomial $f \in \mathsf{F}[x]$, has all coefficients distinct; that is, $f = \sum_{1 \leq i \leq t} c_i x^{e_i}$ and $c_i = c_j \Rightarrow i = j$, then we say f is *diverse*. To interpolate a diverse polynomial $f \in \mathsf{F}[x]$, we first follow the method of Garg and Schost (2009) by computing $f \operatorname{rem} (x^{p_i} - 1)$ for "good primes" p_i such that the sparsity of $f \operatorname{rem} (x^{p_i} - 1)$ is the same as that of f. Since f is diverse, $f \operatorname{rem} (x^{p_i} - 1)$ is also diverse and in fact each modular image has the same set of coefficients.

Algorithm 1: Generic interpolation

```
Input: $\mu \in \mathbb{R}_{>0}$, $T, D, q \in \mathbb{N}$, and a remainder black box for unknown
       $T$-sparse $f \in \mathsf{F}[x]$ with $\deg f < D$
Output: $t \in \mathbb{N}$, $e_1, \ldots, e_t \in \mathbb{N}$, and $c_1, \ldots, c_t \in \mathsf{F}$ such that
        $f = \sum_{1 \le i \le t} c_i x^{e_i}$
1  $t \leftarrow 0$
2  $\lambda \leftarrow \max\left(21, \left\lceil \frac{5}{3} T(T-1) \ln D \right\rceil\right)$
3  for $\lceil \log_2(3/\mu) \rceil$ primes $p \in \{\lambda, \ldots, 2\lambda\}$ do
4      Use black box to compute $f_p = f(x) \operatorname{rem} (x^p - 1)$
5      if $f_p$ has more than $t$ terms then
6          $t \leftarrow$ sparsity of $f_p$
7          $\varrho \leftarrow p$
8  $\alpha \leftarrow$ element of $\mathsf{F}$ such that $\Pr[f(\alpha x)$ is not diverse$] < \mu/3$
9  $g_\varrho \leftarrow f(\alpha x) \operatorname{rem} (x^\varrho - 1)$
10 $c_1, \ldots, c_t \leftarrow$ nonzero coefficients of $g_\varrho$
11 $e_1, \ldots, e_t \leftarrow 0$
12 for $\lceil 2 \ln(3/\mu) + 4(\ln D)/(\ln \lambda) \rceil$ primes $p \in \{\lambda, \ldots, 2\lambda\}$ do
13     Use black box to compute $g_p = f(\alpha x) \operatorname{rem} (x^p - 1)$
14     if $g_p$ has exactly $t$ nonzero terms then
15         for $i = 1, \ldots, t$ do Update $e_i$ with exponent of $c_i$ in $g_p$ modulo
           $p$ via Chinese remaindering
16 for $i = 1, \ldots, t$ do $c_i \leftarrow c_i \alpha^{-e_i}$
17 return $f(x) = \sum_{1 \le i \le t} c_i x^{e_i}$
```

Using this fact, we avoid the need to construct and subsequently factor the symmetric polynomial in the exponents. Instead, we correlate like terms based on the (unique) coefficients in each modular image, then use simple Chinese remaindering to construct each exponent e_i from its image modulo each p_i. This requires only $O(\log d)$ remainder black box evaluations at good primes, gaining a factor of t improvement over the randomized version of Garg and Schost (2009) for diverse polynomials.

In the following sections, we will show how to choose an $\alpha \in \mathsf{F}$ so that $f(\alpha x)$ — which we can easily construct a remainder black box for — is diverse. With such a procedure, Algorithm 1 gives a Monte Carlo algorithm for interpolation over a general field.

Theorem 2.2. *With inputs as specified, Algorithm 1 correctly computes the unknown polynomial f with probability at least $1-\mu$. The total cost in field operations (except for step 8) is*

$$O\left(\ell \cdot \left(\frac{\log D}{\log T + \operatorname{loglog} D} + \log \frac{1}{\mu}\right) \cdot \mathsf{M}\left(T^2 \log D\right)\right).$$

Proof. The for loop on line 3 searches for the true sparsity t and a single good prime ϱ. Since each prime p in the given range is good with probability at least $1/2$ by Lemma 2.1, the probability of failure at this stage is at most $\mu/3$.

The for loop on line 12 searches for and uses sufficiently many good primes to recover the exponents of f. The product of all the good primes must be at least D, and since each prime is at least λ, at least $(\ln D)/(\ln \lambda)$ good primes are required.

Let $n = \lceil 2\ln(3/\mu) + 4(\ln D)/(\ln \lambda)\rceil$ be the number of primes sampled in this loop, and $k = \lceil (\ln D)/(\ln \lambda)\rceil$ the number of good primes required. We can derive that $(n/2-k)^2 \geq (\ln(3/\mu)+k)^2 > (n/2)\ln(3/\mu)$, and therefore $\exp(-2(\frac{n}{2}-k)^2/n) < \mu/3$. Using Hoeffding's Inequality (Hoeffding, 1963), this means the probability of encountering fewer than k good primes is less than $\mu/3$.

Therefore the total probability of failure is at most μ. For the cost analysis, the dominating cost will be the modular black box evaluations in the last for loop. The number of evaluations in this loop is $O(\log(1/\mu) + (\log D)/(\log \lambda))$, and each evaluation has cost $O(\ell \cdot \mathsf{M}(\lambda))$. Since the size of each prime is $\Theta((\log D)/(\log T + \operatorname{loglog} D))$, the complexity bound is correct as stated. □

In case the bound T on the number of nonzero terms is very bad, we could choose a smaller value of λ based on the true sparsity t before line 8, improving the cost of the remainder of the algorithm.

In addition, as our bound on possible number of "bad primes" seems to be quite loose, a more efficient approach in practice would be to replace the for loop on line 12 with one that starts with a prime much smaller than λ and incrementally searches for the next larger primes until the product of all good primes is at least D. We could choose the lower bound to start searching from based on lower bounds on the birthday problem. That is, assuming (falsely) that the exponents are randomly distributed modulo p, start with the least p that will have no exponents collide modulo p with high probability. This would yield an algorithm more sensitive to the true bound on bad primes, but unfortunately gives a worse formal cost analysis.

3 Sparse interpolation over finite fields

We now examine the case that the ground field F is the finite field with q elements, which we denote $\mathbb{F}_q$. First we show how to effectively diversify the unknown polynomial f in order to complete Algorithm 1 for the case of large finite fields. Then we show how to extend this to a Las Vegas algorithm with the same complexity.

3.1 Diversification

For an unknown $f \in \mathbb{F}_q[x]$ given by a remainder black box, we must find an α so that $f(\alpha x)$ is diverse. A surprisingly simple trick works: evaluating $f(\alpha x)$ for a random nonzero $\alpha \in \mathbb{F}_q$.

Theorem 3.1. *For $q \geq T(T-1)T+1$ and any T-sparse polynomial $f \in \mathbb{F}_q[x]$ with $\deg f < D$, if α is chosen uniformly at random from $\mathbb{F}_q^*$, the probability that $f(\alpha x)$ is diverse is at least $1/2$.*

Proof. Let $t \leq T$ be the exact number of nonzero terms in f, and write $f = \sum_{1 \leq i \leq t} c_i x^{e_i}$, with nonzero coefficients $c_i \in \mathbb{F}_q^*$ and $e_1 < e_2 < \cdots < e_t$. So the ith coefficient of $f(\alpha x)$ is $c_i \alpha^{e_i}$.

If $f(\alpha x)$ is *not* diverse, then we must have $c_i \alpha^{e_i} = c_j \alpha^{e_j}$ for some $i \neq j$. Therefore consider the polynomial $A \in \mathbb{F}_q[y]$ defined by

$$A = \prod_{1 \leq i < j \leq t} \left(c_i y^{e_i} - c_j y^{e_j} \right).$$

We see that $f(\alpha x)$ is diverse if and only if $A(\alpha) \neq 0$, hence the number of roots of A over $\mathbb{F}_q$ is exactly the number of unlucky choices for α.

The polynomial A is the product of exactly $\binom{t}{2}$ binomials, each of which has degree less than D. Therefore

$$\deg A < \frac{T(T-1)D}{2},$$

and this also gives an upper bound on the number of roots of A. Hence $q - 1 \geq 2 \deg A$, and at least half of the elements of $\mathbb{F}_q^*$ are not roots of A, yielding the stated result. $\square$

Using this result, given a black box for f and the exact sparsity t of f, we can find an $\alpha \in \mathbb{F}_q$ such that $f(\alpha x)$ is diverse by sampling random values $\alpha \in \mathbb{F}_q$, evaluating $f(\alpha x) \operatorname{rem} x^p - 1$ for a single good prime p, and checking whether the polynomial is diverse. With probability at least $1 - \mu$, this will succeed in finding a diversifying α after at most $\lceil \log_2(1/\mu) \rceil$ iterations. Therefore we can use this approach in Algorithm 1 with no effect on the asymptotic complexity.

3.2 Verification

So far, Algorithm 1 over a finite field is probabilistic of the Monte Carlo type; that is, it may give the wrong answer with some controllably-small probability. To provide a more robust Las Vegas probabilistic algorithm, we require only a fast way to check that a candidate answer is in fact correct. To do this, observe that given a modular black box for an unknown T-sparse $f \in \mathbb{F}_q[x]$ and an explicit T-sparse polynomial $g \in \mathbb{F}_q[x]$, we can construct a modular black box for the $2T$-sparse polynomial $f - g$ of their difference. Verifying that $f = g$ thus reduces to the well-studied problem of deterministic polynomial identity testing.

The following algorithm is due to Bläser et al. (2009) and provides this check in essentially the same time as the interpolation algorithm; we restate it in Algorithm 2 for completeness and to use our notation.

Algorithm 2: Verification over finite fields

```
Input: $T, D, q \in \mathbb{N}$ and remainder black box for unknown $T$-sparse
       $f \in \mathbb{F}_q[x]$ with $\deg f \leq D$
Output: ZERO iff $f$ is identically zero
1 for the least $(T-1)\log_2 D$ primes $p$ do
2     Use black box to compute $f_p = f \operatorname{rem} (x^p - 1)$
3     if $f_p \neq 0$ then return NONZERO
4 return ZERO
```

Theorem 3.2. *Algorithm 2 works correctly as stated and uses at most*

$$O\left(\ell T \log D \cdot \mathsf{M}\left(T \log D \cdot (\log T + \operatorname{loglog} D)\right)\right)$$

field operations.

Proof. For correctness, notice that the requirements for a "good prime" for identity testing are much weaker than for interpolation. Here, we only require that a single nonzero term not collide with any other nonzero term. That is, every bad prime p will divide $e_j - e_1$ for some $2 \leq j \leq T$. There can be $\log_2 D$ distinct prime divisors of each $e_j - e_1$, and there are $T - 1$ such differences. Therefore testing that the polynomial is zero modulo $x^p - 1$ for the first $(T-1)\log_2 D$ primes is sufficient to guarantee at least one nonzero evaluation of a nonzero T-sparse polynomial.

For the cost analysis, the prime number theorem (Bach and Shallit, 1996, Theorem 8.8.4), tells us that the first $(T-1)\log_2 D$ primes are each bounded by $O(T \cdot \log D \cdot (\log T + \operatorname{loglog} D))$. The stated bound follows directly. $\square$

This provides all that we need to prove the main result of this section:

Theorem 3.3. *Given $q \geq T(T-1)D+1$, any $T, D \in \mathbb{N}$, and a modular black box for unknown T-sparse $f \in \mathbb{F}_q[x]$ with $\deg f \leq D$, there is an algorithm that always produces the correct polynomial f and with high probability uses only $O\tilde{~}\left(\ell T^2 \log^2 D\right)$ field operations.*

Proof. Use Algorithms 1 and 2 with $\mu = 1/2$, looping as necessary until the verification step succeeds. With high probability, only a constant number of iterations will be necessary, and so the cost is as stated. $\square$

For the small field case, when $q \in O(T^2 D)$, the obvious approach would be to work in an extension E of size $O(\log T + \log D)$ over $\mathbb{F}_q$. Unfortunately, this would presumably increase the cost of each evaluation by a factor of $\log D$, potentially dominating our factor of T savings compared to the randomized version of Garg and Schost (2009) when the unknown polynomial has very few terms and extremely high degree.

In practice, it seems that a much smaller extension than this is sufficient in any case to make each $\gcd(e_j - e_i, q-1)$ small compared to $q-1$, but we do not yet know how to prove any tighter bound in the worst case.

4 Approximate sparse interpolation algorithms

In this section we consider the problem of interpolating an approximate sparse polynomial $f \in \mathbb{C}[x]$ from evaluations on the unit circle. We will generally assume that f is t-sparse:

$$f = \sum_{1 \leq i \leq t} c_i x^{e_i} \text{ for } c_i \in \mathbb{C} \text{ and } e_1 < \cdots < e_t = d. \tag{4.1}$$

We require a notion of size for such polynomials, and define the coefficient 2-norm of $f = \sum_{0 \leq i \leq d} f_i \, x^i$ as

$$\| f \|_2 = \sqrt{\sum_{0 \leq i \leq d} |f_i|^2}.$$

The following identity relates the norm of evaluations on the unit circle and the norm of the coefficients. As in Section 2, for $f \in \mathbb{C}[x]$ is as in (4.1), we say that a prime p is a *good prime* for f if $p \nmid (e_i - e_j)$ for all $i \neq j$.

Lemma 4.1. *Let $f \in \mathbb{C}[x]$, p a good prime for f, and $\omega \in \mathbb{C}$ a pth primitive root of unity. Then*

$$\|f\|_2^2 = \frac{1}{p} \sum_{0 \le i < p} \left| f(\omega^i) \right|^2 .$$

See Giesbrecht and Roche (2010, Theorem 2.9).

We can now formally define the approximate sparse univariate interpolation problem.

Definition 4.2. *Let $\epsilon > 0$ and assume there exists an unknown t-sparse $f \in \mathbb{C}[x]$ of degree at most D. An ϵ-*approximate black box *for f takes an input $\xi \in \mathbb{C}$ and produces a $\gamma \in \mathbb{C}$ such that $|\gamma - f(\xi)| \le \epsilon |f(\xi)|$.*

That is, the relative error of any evaluation is at most ϵ. As noted in the introduction, we will specify our input points exactly, at (relatively low order) roots of unity. The *approximate sparse univariate interpolation problem* is then as follows: given $D, T \in \mathbb{N}$ and $\delta \ge \epsilon > 0$, and an ϵ-approximate black box for an unknown T-sparse polynomial $f \in \mathbb{C}[x]$ of degree at most D, find a T-sparse polynomial $g \in \mathbb{C}[x]$ such that $\|f - g\|_2 \le \delta \|g\|_2$.

The following theorem shows that t-sparse polynomials are well-defined by good evaluations on the unit circle.

Theorem 4.3. *Let $\epsilon > 0$ and $f \in \mathbb{C}[x]$ be a t-sparse polynomial. Suppose there exists a t-sparse polynomial $g \in \mathbb{C}[x]$ such that for a prime p which is good for both f and $f - g$, and pth primitive root of unity $\omega \in \mathbb{C}$, we have*

$$|f(\omega^i) - g(\omega^i)| \le \epsilon |f(\omega^i)| \quad \text{for } 0 \le i < p.$$

Then $\|f - g\|_2 \le \epsilon \|f\|_2$. Moreover, if $g_0 \in \mathbb{C}[x]$ is formed from g by deleting all the terms not in the support of f, then $\|f - g_0\|_2 \le 2\epsilon \|f\|_2$.

Proof. Summing over powers of ω we have

$$\sum_{0 \le i < p} |f(\omega^i) - g(\omega^i)|^2 \le \epsilon^2 \sum_{0 \le i < p} |f(\omega^i)|^2 .$$

Thus, since p is a good prime for both $f - g$ and f, using Lemma 4.1, $p \cdot \|f - g\|_2^2 \le \epsilon^2 \cdot p \cdot \|f\|_2^2$ and $\|f - g\|_2 \le \epsilon \|f\|_2$.

Since $g - g_0$ has no support in common with f,

$$\|g - g_0\|_2 \le \|f - g\|_2 \le \epsilon \|f\|_2 .$$

Thus

$$\begin{aligned}\|f - g_0\|_2 &= \|f - g + (g - g_0)\|_2 \\ &\le \|f - g\|_2 + \|g - g_0\|_2 \le 2\epsilon \|f\|_2 . \quad \square\end{aligned}$$

□

In other words, any t-sparse polynomial whose values are very close to f must have the same support except possibly for some terms with very small coefficients.

4.1 Computing the norm of an approximate sparse polynomial

Let $0 < \epsilon < 1/2$ and $f \in \mathbb{C}[x]$ a t-sparse polynomial for which we are given an ϵ-approximate black box. We first consider the problem of computing $\|f\|_2$.

Algorithm 3: Approximate norm

```
Input: $T, D \in \mathbb{N}$ and $\epsilon$-approximate black box for unknown $T$-sparse
       $f \in \mathbb{C}[x]$ with $\deg f \le D$
Output: $\sigma \in \mathbb{R}$, an approximation to $\|f\|_2$
1 $\lambda \leftarrow \max\left(21, \left\lceil \frac{5}{3} t(t-1)\ln d \right\rceil\right)$
2 Choose a prime $p$ randomly from $\{\lambda, \ldots, 2\lambda\}$
3 $\omega \leftarrow \exp(2\pi i/p)$
4 $w \leftarrow (f(\omega^0), \ldots, f(\omega^{p-1})) \in \mathbb{C}^p$ computed using the black box
5 return $(1/\sqrt{p}) \cdot \|w\|_2$
```

Theorem 4.4. *Algorithm 3 works as stated. On any invocation, with probability at least $1/2$, it returns a value $\sigma \in \mathbb{R}_{\ge 0}$ such that*

$$(1 - 2\epsilon)\|f\|_2 < \sigma < (1 + \epsilon)\|f\|_2 .$$

Proof. Let $v = (f(\omega^0), \ldots, f(\omega^{p-1})) \in \mathbb{C}^p$ be the vector of exact evaluations of f. Then by the properties of our ϵ-approximate black box we have $w = v + \epsilon\Delta$, where $|\Delta_i| < |f(\omega^i)|$ for $0 \le i < p$, and hence $\|\Delta\|_2 < \|v\|_2$. By the triangle inequality $\|w\|_2 \leq \|v\|_2 + \epsilon\,\|\Delta\|_2 < (1+\epsilon)\,\|v\|_2$. By Lemmas 2.1 and 4.1, $\|v\|_2 = \sqrt{p}\,\|f\|_2$ with probability at least 1/2, so $(1/\sqrt{p}) \cdot \|w\|_2 < (1+\epsilon)\,\|f\|_2$ with this same probability.

To establish a lower bound on the output, note that we can make error in the evaluation relative to the output magnitude: because $\epsilon < 1/2$, $|f(\omega^i) - w_i| < 2\epsilon|w_i|$ for $0 \leq i < p$. We can write $v = w + 2\epsilon\nabla$, where $\|\nabla\|_2 < \|w\|_2$. Then $\|v\|_2 \leq (1+2\epsilon)\,\|w\|_2$, and $(1-2\epsilon)\,\|f\|_2 < (1/\sqrt{p}) \cdot \|w\|_2$. □

4.2 Constructing an ϵ-approximate remainder black box

Assume that we have chosen a good prime p for a t-sparse $f \in \mathsf{F}[x]$. Our goal in this subsection is a simple algorithm and numerical analysis to accurately compute $f \operatorname{rem} x^p - 1$.

Assume that $f \operatorname{rem} x^p - 1 = \sum_{0 \leq i < p} b_i x^i$ exactly. For a primitive pth root of unity $\omega \in \mathbb{C}$, let $V(\omega) \in \mathbb{C}^{p \times p}$ be the Vandermonde matrix built from the points $1, \omega, \ldots, \omega^{p-1}$. Recall that $V(\omega) \cdot (b_0, \ldots, b_{p-1})^T = (f(\omega^0), \ldots, f(\omega^{p-1}))^T$ and $V(\omega^{-1}) = p \cdot V(\omega)^{-1}$. Matrix vector product by such Vandermonde matrices is computed very quickly and in a numerically stable manner by the Fast Fourier Transform (FFT).

Algorithm 4: Approximate Remainder

```
Input: An $\epsilon$-approximate black box for the unknown $t$-sparse
       $f \in \mathbb{C}[x]$, and $p \in \mathbb{N}$, a good prime for $f$
Output: $h \in \mathbb{C}[x]$ such that $\|(f \operatorname{rem} x^p - 1) - h\|_2 \leq \epsilon\,\|f\|_2$.
1 $w \leftarrow (f(\omega^0), \ldots, f(\omega^{p-1})) \in \mathbb{C}^p$ computed using the
      $\epsilon$-approximate black box for $f$
2 $u \leftarrow (1/p) \cdot V(\omega^{-1}) w \in \mathbb{C}^p$ using the FFT algorithm
3 return $h = \sum_{0 \leq i < p} u_i x^i \in \mathbb{C}[x]$
```

Theorem 4.5. *Algorithm 4 works as stated, and*

$$\|(f \operatorname{rem} x^p - 1) - h\|_2 \leq \epsilon\,\|f\|_2\,.$$

It requires $O(p \log p)$ floating point operations and p evaluations of the black box.

Proof. Because f and $f \operatorname{rem} x^p - 1$ have exactly the same coefficients (p is a good prime for f), they have exactly the same norm. The FFT in Step 2 is accomplished in $O(p \log p)$ floating point operations. This algorithm is numerically stable since $(1/\sqrt{p}) \cdot V(\omega^{-1})$ is unitary. That is, assume $v = (f(\omega_0), \ldots, f(\omega^{p-1})) \in \mathbb{C}^p$ is the vector of *exact* evaluations of f, so $\|v - w\|_2 \le \epsilon \|v\|_2$ by the black box specification. Then, using the fact that $\|v\|_2 = \sqrt{p}\,\|f\|_2$,

$$\left\|(f \operatorname{rem} x^{p-1}) - h\right\|_2 = \left\|\frac{1}{p}V(\omega^{-1})v - \frac{1}{p}V(\omega^{-1})w\right\|_2$$
$$= \frac{1}{\sqrt{p}}\left\|\frac{1}{\sqrt{p}}V(\omega^{-1})\cdot(v-w)\right\|_2 = \frac{1}{\sqrt{p}}\|v-w\|_2 \le \frac{\epsilon}{\sqrt{p}}\|v\|_2 = \epsilon\|f\|_2. \quad \square$$

4.3 Creating ϵ-diversity

First, we extend the notion of polynomial diversity to the approximate case.

Definition 4.6. *Let $f \in \mathbb{C}[x]$ be a t-sparse polynomial as in (4.1) and $\delta \ge \epsilon > 0$ such that $|c_i| \ge \delta \|f\|_2$ for $1 \le i \le t$. The polynomial f is said to be* ϵ-diverse *if and only if every pair of distinct coefficients is at least $\epsilon\|f\|_2$ apart. That is, for every $1 \le i < j \le t$, $|c_i - c_j| \ge \epsilon \|f\|_2$.*

Intuitively, if $(\epsilon/2)$ corresponds to the machine precision, this means that an algorithm can reliably distinguish the coefficients of a ϵ-diverse polynomial. We now show how to choose a random α to guarantee ϵ-diversity.

Theorem 4.7. *Let $\delta \ge \epsilon > 0$ and $f \in \mathbb{C}[x]$ a t-sparse polynomial whose nonzero coefficients are of magnitude at least $\delta\|f\|_2$. If s is a prime satisfying $s > 12$ and*

$$t(t-1) \le s \le 3.1\frac{\delta}{\epsilon},$$

then for $\zeta = \mathbf{e}^{2\pi\mathbf{i}/s}$ an s-PRU and $k \in \mathbb{N}$ chosen uniformly at random from $\{0, 1, \ldots, s-1\}$, $f(\zeta^k x)$ is ϵ-diverse with probability at least $\frac{1}{2}$.

Proof. For each $1 \le i \le t$, write the coefficient c_i in polar notation to base ζ as $c_i = r_i\zeta^{\theta_i}$, where each r_i and θ_i are nonnegative real numbers and $r_i \ge \delta\|f\|_2$.

Suppose $f(\zeta^k x)$ is *not* ϵ-diverse. Then there exist indices $1 \le i < j \le t$ such that

$$\left|r_i\zeta^{\theta_i}\zeta^{ke_i} - r_j\zeta^{\theta_j}\zeta^{ke_j}\right| \le \epsilon\|f\|_2.$$

Because $\min(r_i, r_j) \ge \delta\|f\|_2$, the value of the left hand side is at least $\delta\|f\|_2 \cdot \left|\zeta^{\theta_i+ke_i} - \zeta^{\theta_j+ke_j}\right|$. Dividing out $\zeta^{\theta_j+ke_i}$, we get

$$\left|\zeta^{\theta_i-\theta_j} - \zeta^{k(e_j-e_i)}\right| \le \frac{\epsilon}{\delta}.$$

By way of contradiction, assume there exist distinct choices of k that satisfy the above inequality, say $k_1, k_2 \in \{0, \ldots, s-1\}$. Since $\zeta^{\theta_i - \theta_j}$ and $\zeta^{e_j - e_i}$ are a fixed powers of ζ not depending on the choice of k, this means

$$\left| \zeta^{k_1(e_j - e_i)} - \zeta^{k_2(e_j - e_i)} \right| \leq 2\frac{\epsilon}{\delta}.$$

Because s is prime, $e_i \neq e_j$, and we assumed $k_1 \neq k_2$, the left hand side is at least $|\zeta - 1|$. Observe that $2\pi/s$, the distance on the unit circle from 1 to ζ, is a good approximation for this Euclidean distance when s is large. In particular, since $s > 12$,

$$\frac{|\zeta - 1|}{2\pi/s} > \frac{\sqrt{2}\left(\sqrt{3} - 1\right)/2}{\pi/6},$$

and therefore $|\zeta - 1| > 6\sqrt{2}(\sqrt{3} - 1)/s > 6.2/s$, which from the statement of the theorem is at least $2\epsilon/\delta$. This is a contradiction, and therefore the assumption was false; namely, there is at most one choice of k such that the i'th and j'th coefficients collide.

Then, since there are exactly $\binom{t}{2}$ distinct pairs of coefficients, and $s \geq t(t-1) = 2\binom{t}{2}$, $f(\zeta^k x)$ is diverse for at least half of the choices for k. □

We note that the diversification which maps $f(x)$ to $f(\zeta^k x)$ and back is numerically stable since ζ is on the unit circle.

In practice, the previous theorem will be far too pessimistic. We therefore propose the method of Algorithm 5 to adaptively choose s, δ, and ζ^k simultaneously, given a good prime p.

Suppose there exists a threshold $S \in \mathbb{N}$ such that for all primes $s > S$, a random sth primitive root of unity ζ^k makes $f(\zeta^k x)$ ϵ-diverse with high probability. Then Algorithm 5 will return a root of unity whose order is within a constant factor of S, with high probability. From the previous theorem, if such an S exists it must be $O(t^2)$, and hence the number of iterations required is $O(\log t)$.

Otherwise, if no such S exists, then we cannot diversify the polynomial. Roughly speaking, this corresponds to the situation that f has too many coefficients with absolute value close to the machine precision. In this case, we can simply use the algorithm of Garg and Schost (2009) numerically, achieving the same stability but using a greater number of evaluations and bit operations. It is possible to establish an adaptive hybrid between our algorithm and that of Garg and Schost (2009) by making f as ϵ-diverse *as possible* given our precision. The non-zero coefficients of f are clustered into groups which are not ϵ-diverse (i.e., are within $\epsilon \|f\|_2$ of each other). We

Algorithm 5: Adaptive diversification

```
  Input: $\epsilon$-approximate black box for $f$, known good prime $p$, known
         sparsity $t$
  Output: $\zeta, k$ such that $f(\zeta^k x)$ is $\epsilon$-diverse, or FAIL
1 $s \leftarrow 1$,    $\delta \leftarrow \infty$,    $f_p \leftarrow 0$
2 while $s \le t^2$ and $\#\{$coeffs $c$ of $f_s$ s.t. $|c| \ge \delta\} < t$ do
3     $s \leftarrow$ least prime $\ge 2s$
4     $\zeta \leftarrow \exp(2\pi \mathbf{i}/s)$
5     $k \leftarrow$ random integer in $\{0, 1, \ldots, s-1\}$
6     Compute $f_s = f(\zeta^k x) \operatorname{rem} x^p - 1$
7     $\delta \leftarrow$ least number s.t. all coefficients of $f_s$ at least $\delta$ in absolute
      value are pairwise $\epsilon$-distinct
8 if $\delta > 2\epsilon$ then return FAIL
9 else return $\zeta^k$
```

can use the symmetric polynomial reconstruction of Garg and Schost (2009) to extract the exponents within each group.

4.4 Approximate interpolation algorithm

We now plug our ϵ-approximate remainder black box, and method for making f ϵ-diverse, into our generic Algorithm 1 to complete our algorithm for approximate interpolation.

Theorem 4.8. *Let $\delta > 0$, $f \in \mathbb{C}[x]$ with degree at most D and sparsity at most T, and suppose all nonzero coefficients c of f satisfy $|c| > \delta \|f\|_2$. Suppose also that $\epsilon < 1.5\delta/(T(T-1))$, and we are given an ϵ-approximate black box for f. Then, for any $\mu < 1/2$ we have an algorithm to produce a $g \in \mathbb{C}[x]$ satisfying the conditions of Theorem 4.3. The algorithm succeeds with probability at least $1-\mu$ and uses $\tilde{O}(T^2 \cdot \log(1/\mu) \cdot \log^2 D)$ black box evaluations and floating point operations.*

Proof. Construct an approximate remainder black box for f using Algorithm 4. Then run Algorithm 1 using this black box as input. On step 8 of Algorithm 1, run Algorithm 5, iterating steps 5–7 $\lceil \log_2(3/\mu) \rceil$ times on each iteration through the while loop to choose a diversifying $\alpha = \zeta^k$ with probability at least $1 - \mu/3$.

The cost comes from Theorems 2.2 and 4.5 along with the previous discussion and Theorem 4.7. □

Observe that the resulting algorithm is Monte Carlo, but could be made Las Vegas by combining the finite fields zero testing algorithm discussed in Section 3.2 with the guarantees of Theorem 4.3.

5 Implementation results

We implemented our algorithms in C++ using the GNU Multiple Precision Arithmetic Library (GMP, `http://gmplib.org/`) and Victor Shoup's Number Theory Library (NTL, `http://www.shoup.net/ntl/`) for the exponent arithmetic. For comparison with the algorithm of Garg and Schost (2009), we also used NTL's squarefree polynomial factorization routines. We note that, in our experiments, the cost of integer polynomial factorization (for Garg & Schost) and Chinese remaindering were always negligible.

In our timing results, "Determ" refers to the deterministic algorithm as stated in Garg and Schost (2009) and "Alg 1" is the algorithm we have presented here over finite fields, without the verification step. We also developed and implemented a more adaptive, Monte Carlo version of these algorithms, as briefly described at the end of Section 2. The basic idea is to sample modulo $x^p - 1$ for just one prime $p \in \Theta(t^2 \log d)$ that is good with high probability, then to search for much smaller good primes. This good prime search starts at a lower bound of order $\Theta(t^2)$ based on the birthday problem, and finds consecutively larger primes until enough primes have been found to recover the symmetric polynomial in the exponents (for Garg & Schost) or just the exponents (for our method). The corresponding improved algorithms are referred to as "G&S MC" and "Alg 1++" below, respectively.

Table 3 summarizes some timings for these four algorithms over the finite field $\mathbb{Z}/65521\mathbb{Z}$. This modulus was chosen for convenience of implementation, although other methods such as the Ben-Or and Tiwari algorithm might be more efficient in this particlar field since discrete logarithms could be computed quickly. However, observe that our algorithms (and those from Garg and Schost) have only poly-logarithmic dependence on the field size, and so will eventually dominate.

The timings are given in seconds of CPU time on a 64-bit AMD Phenom II 3.2GHz processor with 512K/2M/6M cache, compiled using GCC 4.4.3 with the `-O3` flag. Note that the numbers listed reflect the *base-2 logarithm* of the degree bound and the sparsity bound for the randomly-generated test cases.

The timings are mostly as expected based on our complexity estimates,

$\log_2 D$	T	Determ	G&S MC	Alg 1	Alg 1++
12	10	3.77	0.03	0.03	0.01
16	10	46.82	0.11	0.11	0.08
20	10	—	0.38	0.52	0.33
24	10	—	0.68	0.85	0.38
28	10	—	1.12	2.35	0.53
32	10	—	1.58	2.11	0.66
12	20	37.32	0.15	0.02	0.02
16	20	—	0.91	0.52	0.28
20	20	—	3.5	3.37	1.94
24	20	—	6.59	5.94	2.99
28	20	—	10.91	10.22	3.71
32	20	—	14.83	16.22	4.24
12	30	—	0.31	0.01	0.01
16	30	—	3.66	1.06	0.65
20	30	—	10.95	6.7	3.56
24	30	—	25.04	12.42	9.32
28	30	—	38.86	19.36	13.8
32	30	—	62.53	68.1	14.66
12	40	—	0.58	0.01	0.02
16	40	—	8.98	3.7	1.54
20	40	—	30.1	12.9	8.42
24	40	—	67.97	38.34	16.57
28	40	—	—	73.69	36.24
32	40	—	—	—	40.79

Table 3: Finite Fields Algorithm Timings

Noise	Mean Error	Median Error	Max Error
0	4.440 e−16	4.402 e−16	8.003 e−16
$\pm 10^{-12}$	1.113 e−14	1.119 e−14	1.179 e−14
$\pm 10^{-9}$	1.149 e−11	1.191 e−11	1.248 e−11
$\pm 10^{-6}$	1.145 e−8	1.149 e−8	1.281 e−8

Table 4: Approximate Algorithm Stability

and also confirm our suspicion that primes of size $O(t^2)$ are sufficient to avoid exponent collisions. It is satisfying but not particularly surprising to see that our "Alg 1++" is the fastest on all inputs, as all the algorithms have a similar basic structure. Had we compared to the Ben-Or and Tiwari or Zippel's method, they would probably be more efficient for small sizes, but would be easily beaten for large degree and arbitrary finite fields as their costs are super-polynomial.

The implementation of the approximate algorithm uses machine `double` precision (IEEE), the built-in C++ `complex<double>` type, and the popular Fastest Fourier Transform in the West (FFTW, `http://www.fftw.org/`) package for computing FFTs. Our stability results are summarized in Table 4. Each test case was randomly generated with degree at most 2^{20} and at most 50 nonzero terms. We varied the precision as specified in the table and ran 10 tests in each range. Observe that the error in our results was often *less* than the ϵ error on the evaluations themselves.

Both implementations are released under an MIT-style licence and are available from the second author's website at
`http://www.cs.uwaterloo.ca/~droche/diverse/`.

6 Conclusions

We have shown how to use the idea of diversification to improve the complexity of sparse interpolation over large finite fields by a factor of t, the number of nonzero terms. We achieve a similar complexity for approximate sparse interpolation, and provide the first provably numerically stable algorithm for this purpose. Our experiments confirm these theoretical results.

Numerous open problems remain. A primary shortcoming of our algorithms is the quadratic dependence on t, as opposed to linear in the case of dense interpolation or even sparse interpolation in smaller or chosen finite fields using the Ben-Or and Tiwari algorithm. It seems that reducing this quadratic dependency will not be possible without a different approach, because of the birthday problem embedded in the diversification step. In the approximate case, a provably numerically stable algorithm for sparse interpolation with only $O(t)$ probes is still an open question. And, while general backward error stability is not possible in the high degree case, it would be interesting in the case of low degree and many variables.

Acknowledgements

We thank Reinhold Burger and Éric Schost for pointing out an error in an earlier draft. The comments and suggestions of the anonymous referees were also very helpful, in particular regarding connections to previous results and the proof of Theorem 3.1.

References

Martín Avendaño, Teresa Krick, and Ariel Pacetti. Newton-Hensel interpolation lifting. *Found. Comput. Math.*, 6(1):81–120, 2006.

Eric Bach and Jeffrey Shallit. *Algorithmic number theory. Vol. 1.* Foundations of Computing Series. MIT Press, Cambridge, MA, 1996. Efficient algorithms.

Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proc. STOC'88*, pages 301–309, New York, NY, USA, 1988. ACM.

Markus Bläser, Moritz Hardt, Richard J. Lipton, and Nisheeth K. Vishnoi. Deterministically testing sparse polynomial identities of unbounded degree. *Information Processing Letters*, 109(3):187 – 192, 2009.

John Canny, Erich Kaltofen, and Lakshman Yagati. Solving systems of nonlinear polynomial equations faster. In *Proc. ISSAC'89*, pages 121–128, 1989.

David G. Cantor and Erich Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28:693–701, 1991.

Michael Clausen, Andreas Dress, Johannes Grabmeier, and Marek Karpinski. On zero-testing and interpolation of k-sparse multivariate polynomials over finite fields. *Theoretical Computer Science*, 84(2):151 – 164, 1991.

Annie Cuyt and Wen-shin Lee. A new algorithm for sparse interpolation of multivariate polynomials. *Theoretical Computer Science*, 409(2):180–185, 2008.

Angel Díaz and Erich Kaltofen. On computing greatest common divisors with polynomials given by black boxes for their evaluations. In *Proc. ISSAC'95*, pages 232–239, 1995.

Angel Díaz and Erich Kaltofen. FOXBOX: a system for manipulating symbolic objects in black box representation. In *Proc. ISSAC'98*, pages 30–37, 1998.

Sanchit Garg and Éric Schost. Interpolation of polynomials given by straight-line programs. *Theoretical Computer Science*, 410(27-29):2659 – 2662, 2009.

J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, second edition, 2003.

Mark Giesbrecht and Daniel S. Roche. Detecting lacunary perfect powers and computing their roots. *Journal of Symbolic Computation*, 2010. To appear; preprint at arXiv:1901.1848.

Mark Giesbrecht, George Labahn, and Wen-shin Lee. Symbolic-numeric sparse interpolation of multivariate polynomials. *Journal of Symbolic Computation*, 44(8):943 – 959, 2009.

Dima Yu. Grigoriev, Marek Karpinski, and Michael F. Singer. Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields. *SIAM J. on Computing*, 19(6):1059–1063, 1990.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.*, 58:13–30, 1963.

Sayed Mohammad Mahdi Javadi and Michael Monagan. On factorization of multivariate polynomials over algebraic number and function fields. In *Proc. ISSAC'09*, pages 199–206, 2009.

Seyed Javadi and Michael Monagan. Parallel sparse polynomial interpolation over finite fields. In *Proc. Intl. Wkshp. Parallel and Symbolic Computation (PASCO)*, pages 160–168, 2010.

Seyed Mohammad Mahdi Javadi and Michael Monagan. A sparse modular GCD algorithm for polynomials over algebraic function fields. In *Proc. ISSAC'07*, pages 187–194, 2007.

E. Kaltofen and Z. Yang. On exact and approximate interpolation of sparse rational functions. In *Proc. ISSAC'07*, pages 11–18, 2007. doi: 10.1145/1277548.1277577.

E. Kaltofen, Y. N. Lakshman, and J.-M. Wiley. Modular rational sparse multivariate polynomial interpolation. In *Proc. ISSAC'90*, pages 135–139, New York, NY, USA, 1990. ACM.

E. Kaltofen, Z. Yang, and L. Zhi. On probabilistic analysis of randomization in hybrid symbolic-numeric algorithms. In *Proc. Workshop on Symbolic-Numeric Computation (SNC 2007)*, pages 203–210, 2007. doi: 10.1145/1277500.1277503.

Erich Kaltofen and Wen-shin Lee. Early termination in sparse interpolation algorithms. *J. Symbolic Comput.*, 36(3-4):365–400, 2003.

Erich Kaltofen and Barry M. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *Journal of Symbolic Computation*, 9:301–320, 1990.

Erich Kaltofen and Lakshman Yagati. Improved sparse multivariate polynomial interpolation algorithms. In *Proc. ISSAC'88*, pages 467–474, 1989.

Erich Kaltofen, John P. May, Zhenfeng Yang, and Lihong Zhi. Approximate factorization of multivariate polynomials using singular value decomposition. *Journal of Symbolic Computation*, 2008.

Erich L. Kaltofen. Fifteen years after DSC and WLSS2: What parallel computations I do today. In *Proc. Intl. Wkshp. Parallel and Symbolic Computation (PASCO)*, pages 10–17, 2010. doi: 10.1145/1837210.1837213.

Y. Mansour. Randomized approximation and interpolation of sparse polynomials. *SIAM Journal on Computing*, 24(2):357–368, 1995.

J. M. Pollard. Monte Carlo methods for index computation (mod p). *Math. Comp.*, 32(143):918–924, 1978. doi: 10.1090/S0025-5718-1978-0491431-9.

J. M. Pollard. Kangaroos, monopoly and discrete logarithms. *Journal of Cryptology*, 13:437–447, 2000. doi: 10.1007/s001450010010.

J. B. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Ill. J. Math.*, 6:64–94, 1962.

Nitin Saxena. Progress on polynomial identity testing. *Bull. EATCS*, 99: 49–79, 2009.

Andrew J. Sommese, Jan Verschelde, and Charles W. Wampler. Numerical decomposition of the solution sets of polynomial systems into irreducible components. *SIAM Journal on Numerical Analysis*, 38(6):2022–2046, 2001.

Andrew J. Sommese, Jan Verschelde, and Charles W. Wampler. Numerical factorization of multivariate complex polynomials. *Theoretical Computer Science*, pages 651–669, 2004.

Hans J. Stetter. *Numerical Polynomial Algebra.* SIAM, 2004.

Richard Zippel. Interpolating polynomials from their values. *Journal of Symbolic Computation*, 9(3):375 – 403, 1990.